

Draw It or Lose It 2

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 1/28/2024 | Joseph Thomas | Initial |

**Instructions**

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## [Executive Summary](#_sbfa50wo7nsh)

Draw It or Lose It 2 is a web-based game based on the original Android game.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

Web-based game may have design constraints due to browser compatibility.

## [System Architecture View](#_ilbxbyevv6b6)

Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

## [Domain Model](#_8h2ehzxfam4o)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

The GameService, Game, Team, and Player classes are all encapsulated and the Entity class inherits from these classes.

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client’s requirements and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Hardware is too expensive for server, and no dedicated server operating system software. ARM architecture may make development harder of a server application. | Has the best stability of the different operating systems for server use because of minimal environment. Lacks the ease of use compared to others. Can pick and choose hardware needed. | Easy to use, mostly stable, but need to have a Windows Server license which can be expensive. The operating system software is also closed source and may not work well with other platforms. | Should not be able to host a server. It lacks the environment and physical hardware (mouse, keyboard) to do so and primarily uses a touch screen. |
| **Client Side** | The game should have no problem running on Mac platform since it is a Java app. May require more hardware for testing which can be expensive. Testing may also take time to debug if unexpected issues occur. | Linux should not be your primary concern, as Linux has a very minimal gaming community. Since Java is mostly cross platform, as long as the Java Runtime Environment is installed, it should mostly work, but more development time may be needed for testing. | This should be the target market. Most testing and development time should be spent on this side of the client for non-mobile devices. | This should be a great market to cater to. The touch screen interface may require more development time to differentiate the input methods. |
| **Development Tools** | Eclipse and Java IDE should be all that is needed to develop. | Eclipse and Java IDE should be all that is needed to develop. | Eclipse and Java IDE should be all that is needed to develop. | Android Studio should be used for Android app. iOS app is not recommended in Java. May need to port project to C++ and Xcode in MacOS for an iOS app. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

1. **Operating Platform**: I would recommend the Windows operating platform to expand Draw It or Lose It. This would be a great way to expand the user base of the game as Windows is one of the most dominant client platforms in the world.
2. **Operating Systems Architectures**: The Windows 11 platform uses an NT-based kernel with a GUI. The GUI interface supports Eclipse IDE and the Java language allowing easier porting.
3. **Storage Management**: I would recommend storing the files locally until the application is started. While the application starts, load all necessary components into RAM for faster access.
4. **Memory Management**: The local storage will be used, but once the application starts we will load the program into RAM and store it there until the application exits. We will store any progress in the server.
5. **Distributed Systems and Networks**: The client data progress can be uploaded to the server and stored there safely. No identifying information will be stored other than game progress. If game progress fails to upload for any reason, the user will start at their last point of progress.
6. **Security**: The client will need to provide a username and password for a valid account stored on the server. The server will run Linux and have increased security measures.